The ALICE ITS geometry in sPHENIX

Tony Frawley
FSU

May 15, 2016

Importing the ALICE ITS staves into sPHENIX

Following the effort in Santa Fe by Kun, Jin and Darren to make a .gdml file that can be imported into the sPHENIX G4 simulation, I have set up the code needed to implement it.

https://github.com/adfrawley/coresoftware/tree/ITS_MAPS_development/
PHG4MapsSubsystem.(h,cc)
PHG4MapsDetector.(h,cc)
PHG4MapsSteppingAction.(h,cc)
PHG4CylinderGeom_MAPS.(h,cc)
PHG4CylinderCell_MAPS.(h,cc)
PHG4MapsCellReco.(h,cc)
PHG4CylinderCellGeom.(h,cc)

https://github.com/adfrawley/macros/tree/ITS_MAPS_development/macros/g4simulations
G4_ITS_MAPS.C

We import only the staves

The idea is to import the staves for each layer into the sPHENIX simulation setup, and position them by specifying:

- Layer radius
- Number of staves per layer (implies ϕ angle step)
  - If the radius changes, use the same arc length as in the ITS
- Stave tilt (my guess for now)

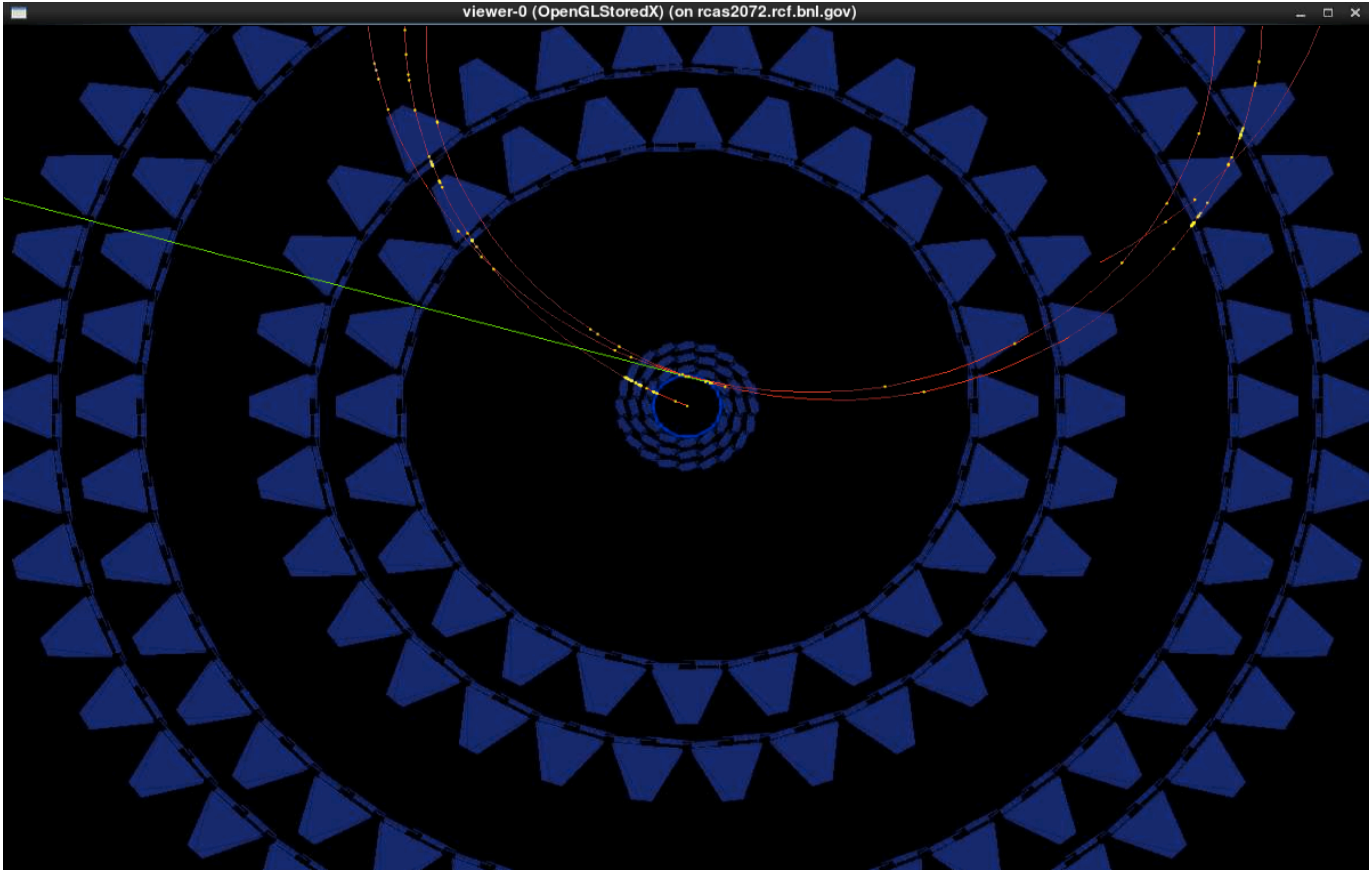
It is really that simple.

The next few slides show some end-views of the resulting tracker, with a single electron thrown in each case

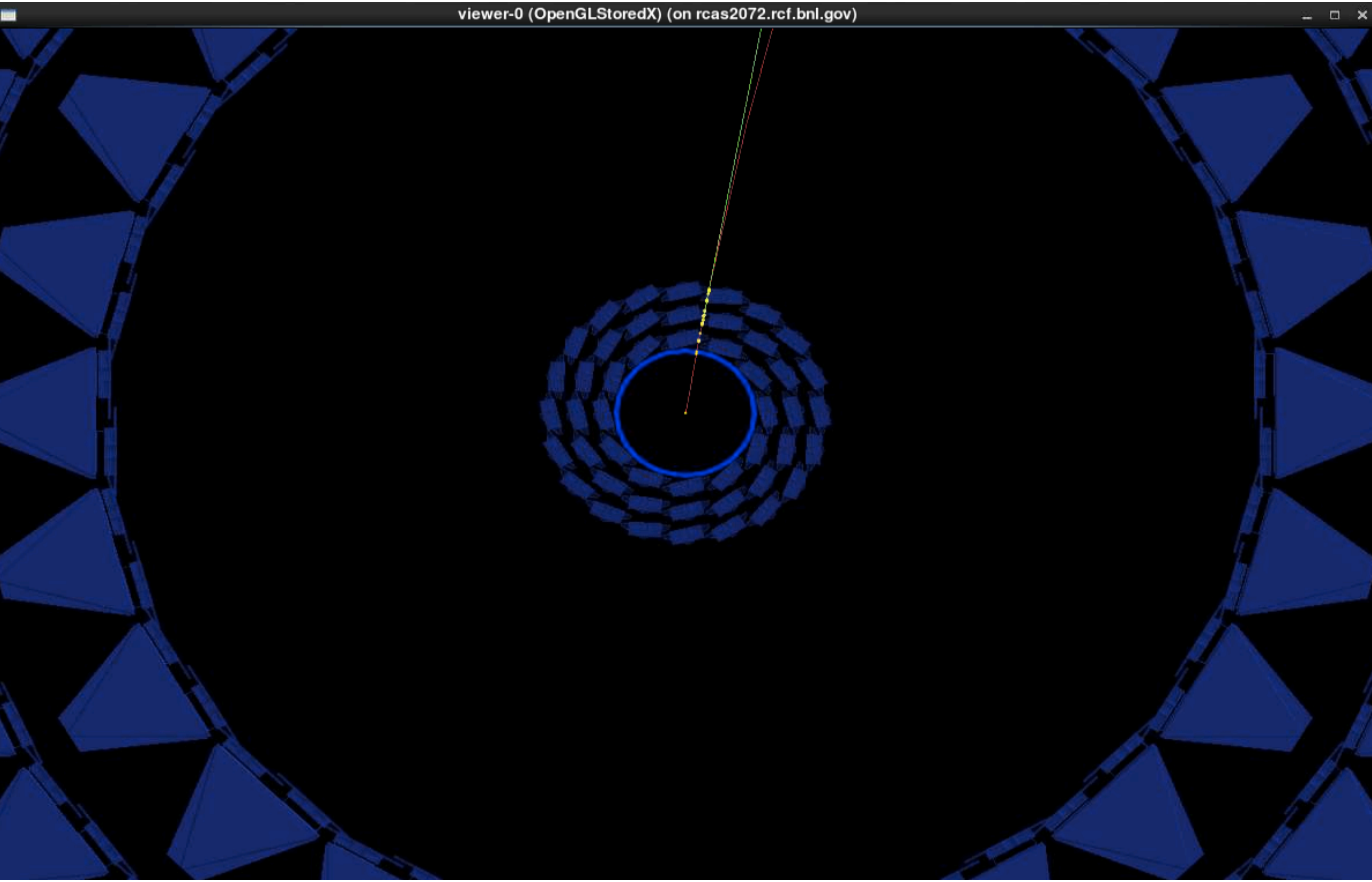
These are for the nominal radii used in the ITS:

```
double maps_layer_radius[7] = {23.0, 31.0, 39.0, 194.0, 247.0, 353.0, 405.0};   // mm
```

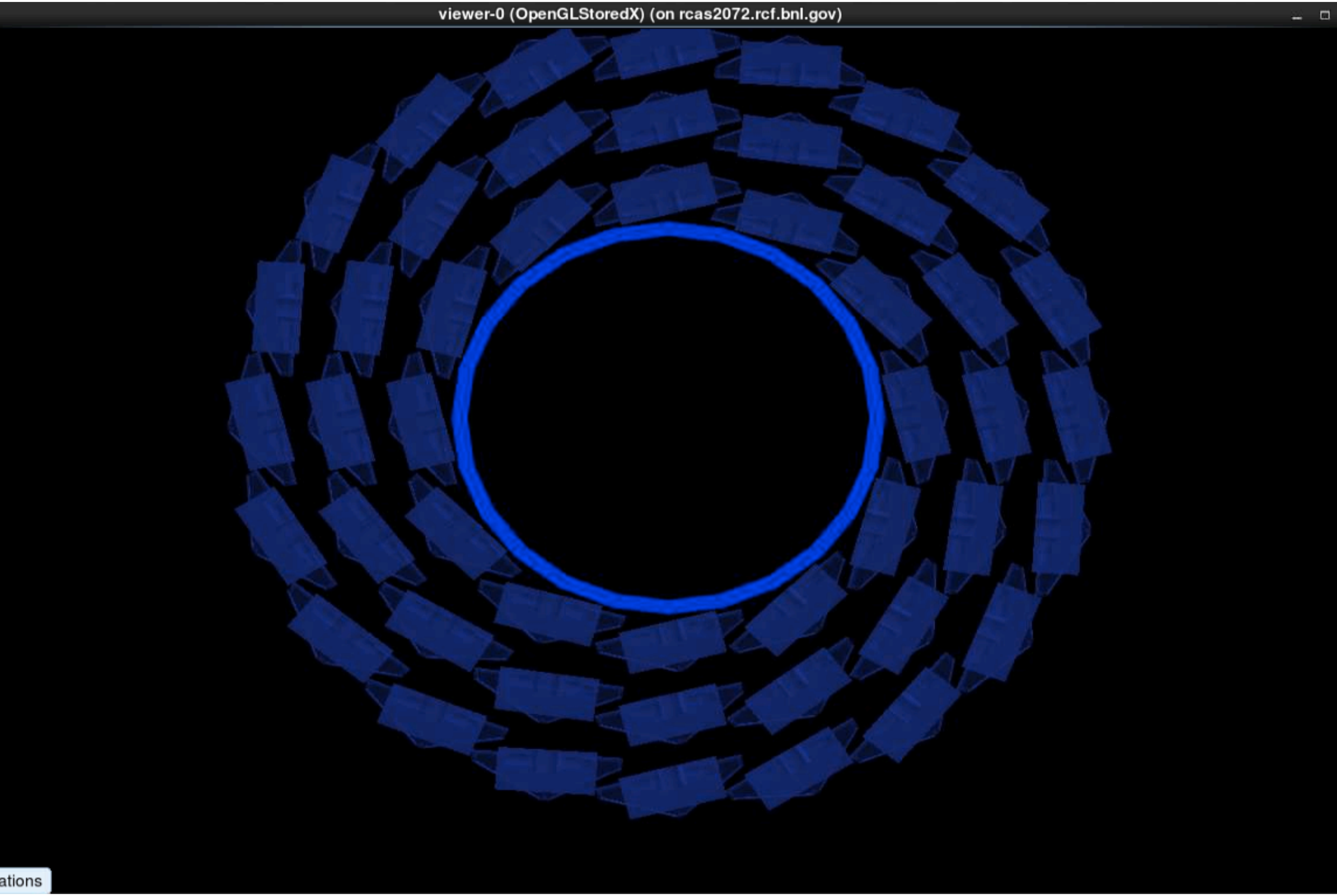
The ALICE ITS staves in sPHENIX



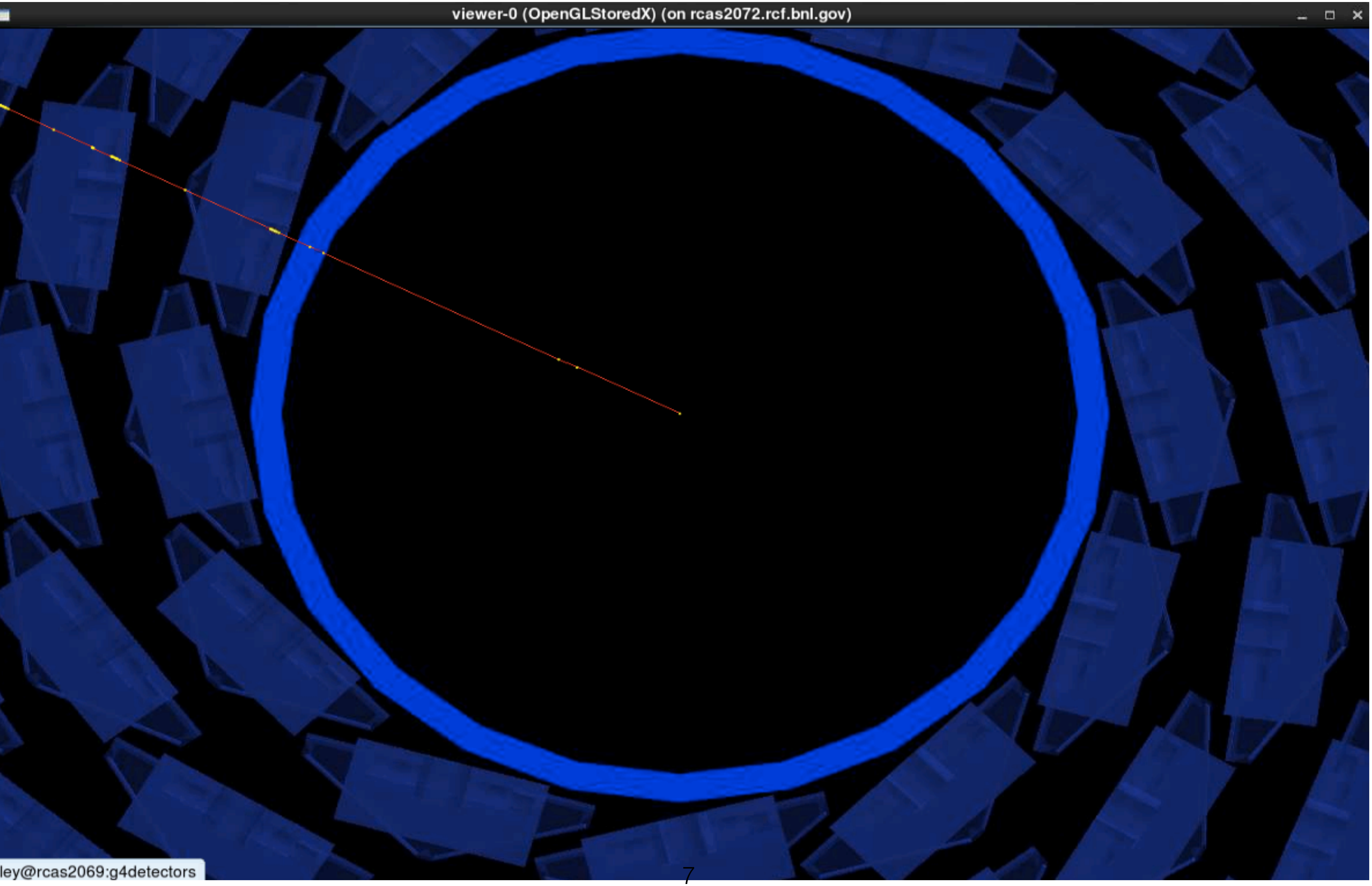
The ALICE ITS staves in sPHENIX



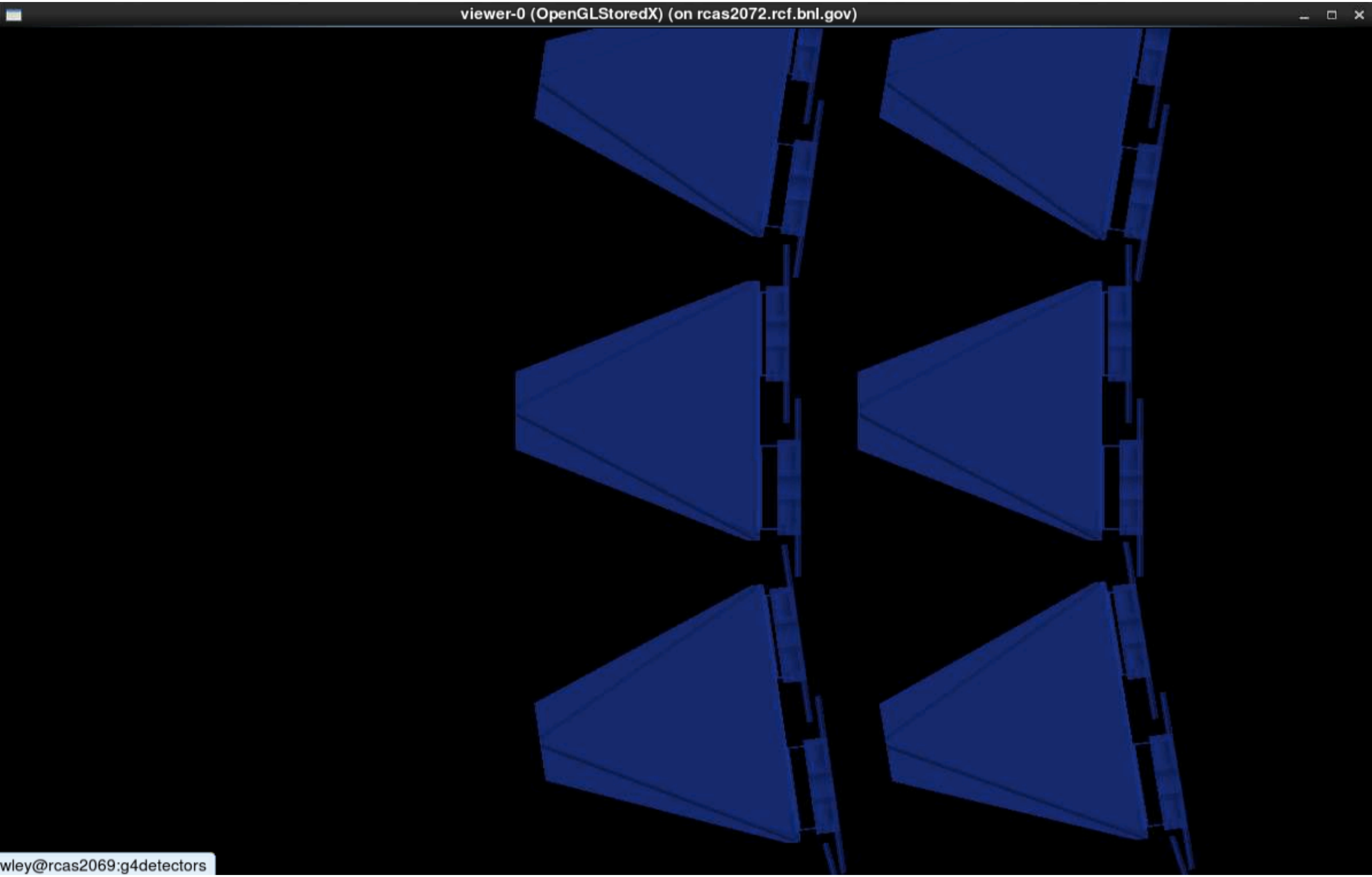
The ALICE ITS staves in sPHENIX



The ALICE ITS staves in sPHENIX



The ALICE ITS staves in sPHENIX



viewer-0 (OpenGLStoredX) (on rcas2072.rcf.bnl.gov)

wley@rcas2069:g4detectors

Tilt angle

The tilt angle used for the inner barrel is 0.25 radians (from eye-balling the display - we need to find out what tilt ALICE is planning to use).

- The tilt angle for layers 3-6 is zero at present, but if you look carefully at the display, some tilt may be better.

Hits storage

I have implemented the hits in PHG4Hit:

```
 UserSteppingAction: layer 6 chip  9 module 3 stave 22 half_stave 0 edep = 3.61892e-06
Particle: e-
  stepping action found hit:
New Hitv1  0x600000000000002  on track 1 EDep 3.61892e-06
Location: X -35.5215/-35.5231  Y 14.856/14.8567  Z 4.339/4.33915
Time        1.28876/1.28882
       10:  px in =   -8.25659
       11:  px out = -8.25661
       12:  py in =   3.53841
       13:  py out = 3.53835
       14:  pz in =   0.745312
       15:  pz out = 0.745372
       101: layer ID =      6
       114: stave index =       22
       115: half stave index =       0
       116: module index =    3
       117: chip index =  9
       118: local x pos in =    -0.403366
       119: local y pos in =    0.0009
       120: local z pos in =    -1.329
       125: local x pos out =   1.15834
       126: local y pos out =   0.005
       127: local z pos out =   4.33915
```

The local entry and exit position in the sensor is derived from the global hit position via a G4 transformation into the frame of the sensor volume.

The geometry object

The geometry object is not quite finished yet.

At present, the hits object records:

- Stave number
- Half-stave number
- Module number
- Chip number

where chip number is equivalent to sensor number (they are 1-1).

But the simulation does not include the pixels, so we determine the hit pixel positions from:

- The address of the sensor => sensor center
- The positions of the entry and exit hit positions in local coordinates => location of hit relative to sensor center

This is largely implemented, but I am still chasing a bug in the transformation of the position in the local sensor frame back to the world (needed to get the pixel location in the world).

This is presently being exercised in PHG4MapsCellReco.

What remains to be done?

Finish the geometry object so that the hit positions in the sensor are available, then:

Comments in the macro G4_ITS_MAPS.C:

// still need to digitize the pixel energy

// still need to apply live area efficiency to hits

// still need to apply MIP threshoilds to hits

// still need to make clusters

// still need to reconstruct tracks

// still need to run ghost rejection

// still need to make track projections

// still need to run beam spot reco